Tarea 6

Introducción y Taller de programación
I-Semestre 2022
Yarman Charpentier
Manfred Jones
Marco Rodríguez

12 de mayo de 2022

1. Sección 6.6.1. Vectores

1.1. Máximo

Listing 1: Esta función recibe un vector y retorna un vector de dos posiciones con el valor máximo y su posición

```
def mayor(x, y):
    return x >= y

def m ximo(vector):
    max = vector[0]
    pos = 1

    count = 0
    while count < len(vector)-1:
        if mayor(vector[count + 1], max):
            max = vector[count + 1]
            pos = vector.index(vector[count + 1]) + 1
        count += 1
    return [max, pos]
```

1.2. Mínimo

```
def menor(x, y): #preguntar por el resultado
    return x <= y

def minimo(vector):
    min = vector[0]
```

```
    pos = 1

    cont = 0
    while cont < len(vector)-1:
        if menor(vector[cont + 1], min):
            min = vector[cont + 1]
            pos = vector.index(vector[cont + 1]) + 1
        cont += 1
    return [min, pos]
```

1.3. Multiplicar escalar por vector

Listing 2: Esta función recibe un número y un vector y retorna el vector con los valores multiplicados por el número

```
def multiplicar(n, vector):
    for j in range(len(vector)):
        vector[j-1] *= n
    return vector
```

1.4. Suma de vectores

```
def nueva_matriz(cant_fil,cant_col):
    nueva_matriz = [[0]] * cant_fil

    for pos_fil in range(cant_fil):
        nueva_matriz[pos_fil] = nueva_matriz[pos_fil] * cant_col

    return nueva_matriz
def suma_matriz(matriz1, matriz2):
    filas = len(matriz1)
    cols = len(matriz1[0])
    matriz_vacia = nueva_matriz(filas,cols)

    for posFila in range(filas):
        for posCol in range(cols):
            matriz_vacia[posFila][posCol] = matriz1[posFila][posCol] +
            matriz2[posFila][posCol] #es parte de la linea anterior

    return matriz_vacia
```

1.5. Producto punto

Listing 3: Esta función recibe dos vectores y retorna su producto punto

```
def productopuntoAux(vector1, vector2, suma):
    if len(vector1) < 1:
        return suma

    suma += vector1[0] * vector2[0]
    return productopuntoAux(vector1[1:], vector2[1:], suma)


def productopunto(vector1, vector2):
    suma = 0
    return productopuntoAux(vector1, vector2, suma)
```

1.6. Solución de un polinomio

```
def cero_de_polinomios(a,b,c,d):
    ceros_de_d=[]
    ceros_de_a=[]
    posibles_raices=[]
    for i in range(-d,d+1):
        if i==0:
            continue
        if d%i==0:
            ceros_de_d.append(i)
    for j in range(-a, a + 1):
        if j == 0:
            continue
        if a % j == 0:
            ceros_de_a.append(j)
    print(ceros_de_d)
    print(ceros_de_a)

    for k in ceros_de_d:
        for m in ceros_de_a:

                posibles_raices.append(k/m)
    return list(set(posibles_raices))
#print(cero_de_polinomios(1,-5,2,8))
#Devuelve [1.0, 2.0, 4.0, 8.0, -1.0, -8.0, -4.0, -2.0]
```

1.7. Distancia entre vectores

Listing 4: Esta función recibe dos vectores y retorna la distancia entre ambos

```
import math
```

```
def distanciaAux(v, w, n):
    suma = 0

    for i in range(1, n+1):
        suma += math.pow((v[i-1] - w[i-1]), 2)

    dist = math.sqrt(suma)

    return dist


def distancia(v, w):
    n = len(v)
    return distanciaAux(v, w, n)
```

1.8. Vector a g grados de otro vector

```
import math


def vector(vector,n):
    radianes_n=n*(math.pi/180)
    rotaciones=[[math.cos(radianes_n),-1*(math.sin(radianes_n))],[math.sin(radia
    #rotaciones=[[math.cos(radianes_n),-1*(math.sin(radianes_n))],
    #[math.sin(radianes_n),math.cos(radianes_n)]]


    vector_rotado=[]
    for i in range(len(vector)):
        vector_rotado+=multiplicacion_de_matrices([vector[i]],rotaciones)

    return vector_rotado

def matriz_vacia(n,m):
    res=[]
    for i in range(n):
        fila=[]
        for j in range(m):
            fila.append(0)
        res.append(fila)
    return res

def multiplicacion_de_matrices(a, b): #producto punto
    fil = len(a)
```

```
    col = len(b[0])
    resp = matriz_vacia(fil, col)
    for i in range(fil):
        for j in range(col):
            for k in range(len(b)):
                resp[i][j] += a[i][k] * b[k][j]
    return resp
#matriz1=[[1,0],[3,4]]


#print(vector(matriz1,30))
#Devuelve [[0.8660254037844387, -0.4999999999999994],
#[4.598076211353316, 1.964101615137755]]
```

---

2. Sección 6.6.2. Matrices

2.1. Exponenciación de matrices (cuadradas)

```
def matriz_vacia(n,m):
    res=[]
    for i in range(n):
        fila=[]
        for j in range(m):
            fila.append(0)
        res.append(fila)
    return res
def multiplicacion_de_matrices(a,b):
    fil=len(a)
    col = len(b[0])
    resp= matriz_vacia(fil,col)
    for i in range (fil):
        for j in range(col):
            for k in range(len(b)):
                resp[i][j]+=a[i][k]*b[k][j]
    return resp

def exponenciacion_matrices(matriz,n):
    if n==1:
        return matriz
    resp=exponenciacion_matrices(matriz,n//2)
    print(resp)
    resp=multiplicacion_de_matrices(resp,resp)
    if n%2:
```

```
        resp=multiplicacion_de_matrices(matriz,resp)
    print(resp)
    return resp
```

3. Sección 6.6.3. Listas

3.1. Ordenar una lista

```
def ordenar_lista(lista):
    for i in range(0,len(lista)):
        for j in range(i+1,len(lista)):
            if(lista[i]>lista[j]):
                lista[i],lista[j] = lista[j],lista[i]
    return lista
```

3.2. Encontrar una sublista desde la posición i hasta la posición j

Listing 5: Esta función recibe una lista y dos valores i y j, y retorna la sublista desde la posición i hasta la j

```
def sublista(list, i, j):
    list = list[i-1:]

    list = list[:(j-i) + 1]

    return list
```

3.3. Buscar la cantidad de elementos múltiplos de k en una lista de números

```
def es_multiplo(numero, multiplo):
    return numero % multiplo == 0
def multiplos(lista,multiplo):
    cont = 0
    for i in range(len(lista)):
        numero = lista.pop(0)
        if es_multiplo(numero,multiplo) == True:
            cont += 1
        else:
            continue
    return cont
```

3.4. Lista de los primeros n enteros

Listing 6: Esta función recibe un entero n y retorna los primeros n enteros

```
def primerosEnteros(n):
    list = []
    for i in range(1, n+1):
        list.append(i)
    return list
```

3.5. Primeros primos

```
def primeros_primos(n):
    primos = []
    num = 0
    while len(primos) < n:
        if num > 1:
            for i in range(2, num):
                if (num % i) == 0:
                    break
            else:
                primos.append(num)
        num += 1
    return primos
```

3.6. Intercambiar dos elementos

Listing 7: Esta función recibe dos enteros i j y una lista, y retorna la lista con los elementos de las posiciones i y j intercambiados

```
def intercambiar(i, j, list):
    temp = list[i-1]
    list[i-1] = list[j-1]
    list[j-1] = temp
    return list
```

3.7. Reemplazar elementos

```
lista = []
#La lista debe estar compuesta por strings
viejo_elemento = input()
nuevo_elemento = input()
def Cambio_elementoDeLista(lista,viejo_elemento,nuevo_elemento):
    for x in range(len(lista)):
        if lista[x] == str(viejo_elemento):
```

```
            lista[x] = str(nuevo_elemento)
        return lista
```

3.8. Eliminar duplicados

Listing 8: Esta función recibe una lista y la retorna sin duplicados

```
def duplicadosAux(list, new_list):
    for i in list:
        if i not in new_list:
            new_list.append(i)

    return new_list

def duplicados(list):
    new_list = []
    return duplicadosAux(list, new_list)
```

3.9. Es Palíndromo

```
def es_palindromo(lista): #Tomar en cuenta que el ejemplo sea un
    for i in range(len(lista)): #palindromo tipo espejo.
        lista[i] = lista[i].lower()

    for j in range(len(lista)):
        if lista[j] != lista[-j-1]:

            return False
    return True
```

3.10. Permutaciones

Listing 9: Esta función recibe una lista y retorna todas sus posibles permutaciones

```
def permutaciones(list, res = []):
    if len(list) == 0:
        print(res)
    for i in range(len(list)):
        permutaciones(list[:i] + list[i + 1:], res + list[i:i + 1])
```